

# Введение

В этой статье мы рассмотрим задачу нахождения остовного дерева минимальной степени для графа  $G$  (MDST Problem) - это такое остовное дерево, что максимальная степень вершины в нём минимальна.

Данная задача является  $NP$ -полной

Мы рассмотрим алгоритм аппроксимации, который за полиномиальное время находит остовное дерево степени не больше  $\Delta + 1$  (где  $\Delta$  - степень некоторого оптимального остового дерева)

## 1 Введение

*Остовное дерево* неориентированного связного графа  $G$  - подграф, который является деревом и в котором присутствуют все вершины графа  $G$ .

*Остовное дерево минимальной степени* неориентированного связного графа  $G$  - такое остовное дерево графа  $G$ , что максимальная степень вершины в этом дереве минимальна среди всех остовных деревьев  $G$ .

Пусть дан граф  $G$  и число  $k \geq 2$ .

Задача проверки существования остовного дерева графа  $G$ , степени не больше  $k$  - является  $NP$ -полной.

Мы не будем это доказывать, но для  $k = 2$  эта задача эквивалентна поиску гамильтонова пути в графе.

### 1.1 Применение

Нам нужна система, которая умеет распространять новости или почту по всему интернету, чтобы каждый человек в мире мог её прочесть. Понятно, что информация передаётся от сервера к серверу, но при этом каждый сервер хочет уменьшить на себя нагрузку. Один из способов достичь этого - построить остовное дерево минимальной степени для сети интернета и распространять по нему информацию.

Ещё одно применение - проектирование энергетической системы. Для энергостанции расщеплять выходную энергию на несколько потоков -

дорогая операция, цена которой быстро растёт со степенью расщепления. Таким образом, правильное проектирование может существенно повысить эффективность энергосистемы.

## 2 Алгоритм $\Delta + 1$

Мы рассмотрим аппроксимирующий алгоритм, который для графа  $G = (V, E)$  находит остовное дерево степени не больше  $\Delta + 1$  (где  $\Delta$  - степень некоторого оптимального остового дерева)

### 2.1 Идея

Для начала найдём произвольное остовное дерево  $T$  графа  $G$ . Обозначим за  $p(u)$  - степень вершины  $u$  в дереве  $T$

Идея алгоритма: Мы попытаемся итеративно уменьшить степени некоторых вершин, которые имеют большую степень в дереве  $T$ .

Наш алгоритм будет работать итерационно.

Пусть  $k$  - максимальная степень вершины в дереве  $T$

Обозначим  $S_k = \{u : p(u) = k\}$  - множество вершин степени  $k$

На каждом шаге (итерации) мы попытаемся уменьшить размер множества  $S_k$  на 1, при этом степени других вершин могут увеличиться, но не больше, чем до  $k - 1$ . Если нам это удастся, то мы рекурсивно перейдём к следующему шагу. Если нет, то это будет означать, что найденное дерево и является ответом. (Это будет доказано чуть позже)

Заметим, что наш алгоритм не будет работать бесконечно. Оценим число шагов в алгоритме:

Пусть  $n$  - число вершин в графе

В дереве не может быть больше  $\frac{2(n-1)}{k}$  вершин степени  $k$  и на каждой итерации это число уменьшается на 1  $\Rightarrow$  число шагов, когда максимальная степень вершины дерева  $T$  равна  $k$  -  $O(n/k)$ . Просуммируем гармонический ряд и получим, что всего шагов в алгоритме  $O(n \log n)$ , т. к. на каждом шаге  $k$  будет либо остваться таким же либо уменьшаться.

## 2.2 Шаг алгоритма

Рассмотрим ребро  $(u, v) \in G \setminus T$ . Пусть  $C$  - уникальный цикл, который появится, если в граф  $T$  добавить ребро  $(u, v)$ . Пусть есть вершина  $w \in C$  - на цикле, для которой выполнено  $p(w) \geq \max(p(u), p(v)) + 2$ . Назовём "улучшением" в  $T$  - добавление ребра  $(u, v)$  и удаление одного из рёбер  $C$ , инцидентного  $w$ . Мы называем это улучшением, потому что  $\max(p(u), p(v), p(w))$  уменьшился хотя бы на 1.

На каждом шаге алгоритма мы хотим попытаться цепочкой таких улучшений уменьшить число вершин максимальной степени  $k$ . В процессе этого степени других вершин могут увеличиваться до тех пор, пока их степень меньше  $k$

Используя предыдущие обозначения и  $p(w) = k$ , скажем, что Вершина  $u$  блокирует вершину  $w$  от ребра  $(u, v)$ , если  $p(u) \geq k - 1$ . Если же ни  $u$  ни  $v$  не блокируют вершину  $w$ , то ребро  $(u, v)$  можно использовать, чтобы уменьшить степень  $w$  через шаг "улучшения".

**Теорема 1.** Пусть  $T$  - остовное дерево степени  $k$  графа  $G$ . Пусть  $\Delta$  - степень остовного дерева минимальной степени графа  $G$ . Пусть  $S_k = \{u : p(u) = k\}$ . Пусть  $B \subset S_{k-1} = \{u : p(u) = k - 1\}$  - подмножество вершин степени  $k - 1$ . Удалим вершины  $S_k \cup B$  из графа. Тогда дерево  $T$  разобьётся на лес  $F$ . Тогда, если в графе  $G$  нет рёбер между разными компонентами связности леса  $F$ , то  $k \leq \Delta + 1$ . (То, чего мы добиваемся)

*Док-во.* Т. к. в графе  $G$  нет рёбер между разными деревьями леса  $F$ , то в любом остовном дереве, нам нужно будет связать эти деревья через вершины в  $S_k \cup B$ .

$F$  содержит хотя бы  $\text{cnt} = |S_k|k + |B|(k - 1) - 2(|S_k| + |B| - 1)$  поддеревьев. (Мы посчитали рёбра, инцидентные вершинам степени  $k$  и  $k - 1$  и вычли рёбра между ними).

Следовательно в любом остовном дереве графа  $G$ , средняя степень вершины из подмножества  $S_k \cup B$  - хотя бы  $\frac{\text{cnt} + (|S_k| + |B| - 1)}{|S_k| + |B|} = k - 1 - \frac{|B| - 1}{|S_k| + |B|}$ .

Вершина с максимальной степенью имеет хотя бы степень  $\Rightarrow \Delta \geq k - 1$

□

### Идея шага алгоритма:

Рассмотрим  $S_k$  - множество вершин максимальной степени дерева  $T$ . Предположим, что мы удалили  $S_k$  из графа и у нас получился лес  $F$ .

- 1 Если в графе  $G$  нет рёбер между компонентами из  $F$ , то можно воспользоваться теоремой 1, где  $B = \emptyset$  и получить даже лучшую оценку, что  $k = \Delta$ . Заканчиваем алгоритм.
- 2 Если в графе есть хотя бы одно ребро  $e = (u, v)$  между разными компонентами  $F$  и оно незаблокировано (т. е.  $\deg(u) \leq k - 2 \wedge \deg(v) \leq k - 2$ ), тогда можно применить улучшение и уменьшить  $|S_k|$  на 1. Переходим к новому шагу алгоритма.
- 3 Иначе - в графе есть хотя бы одно ребро  $e = (u, v)$  между разными компонентами  $F$ , но все они заблокированы (либо через  $u$ , либо через  $v$ , либо через обе вершины).  
Пусть  $w$  - вершина степени  $k$ ,  $u$  -

A graph is chordal if every cycle of length at least 4 has a chord. Here are some files of connected chordal graphs.

4 vertices (5) 5 vertices (15) 6 vertices (58) 7 vertices (272) 8 vertices (1614) 9 vertices (11911) 10 vertices (109539) 11 vertices (1247691) 12 vertices (17566431, gzipped) 13 vertices (305310547, gzipped) вершина, которая блокирует вершину  $w$ , через ребро  $(u, v)$ . Заметим, что тогда  $\deg(u) = k - 1$ . Рассмотрим  $F_u$  - компонента связности из  $F$ , которая содержит  $u$ .

Предположим, что степень вершины  $u$  можно уменьшить на 1, если запустить эту же процедуру на подграфе, индуцированном вершинами  $F_u$ . Тогда вершину  $u$  можно сделать неблокирующей и уже цепочка улучшений уменьшает степень вершины  $w$ .

#### **Замечание.**

В предыдущем абзаце, чтобы улучшить вершину  $u$ , нам достаточно искать улучшение в подграфе индуцированным  $F_u$ , потому что если есть улучшение для вершины  $u$ , которое содержит в себе рёбра не из  $F_u$ , то оно содержит в себе ребро, инцидентное вершине степени  $k$  и можно сразу улучшить вершину из  $S_k$

#### **Реализация шага алгоритма:**

Итак, сам алгоритм:

Пусть  $G$  - исходный граф.  $T$  - какое-то его остовное дерево. Будем итеритивно делать шаги алгоритма:

Удалим вершины  $S_k \cup S_{k-1}$  из дерева  $T$  и пометим все оставшиеся компоненты

связности в дереве  $T$  как "хорошие". Все вершины  $S_k \cup S_{k-1}$  пометим как "плохие".

- 1 Если в графе  $G$  нет рёбер между разными хорошими компонентами, то заканчиваем алгоритм.
- 2 Иначе - в графе  $G$  есть ребро  $e = (u, v) \in G \setminus T$  между двумя хорошими компонентами  $F_u$  и  $F_v$ . Добавим ребро  $(u, v)$  в граф  $T$ . Пусть  $C$  - уникальный цикл, который образовался в графе  $T$ .

Если на цикле есть вершина степени  $k$ , то мы нашли цепочку улучшений для вершины степени  $k$ . Применяем её и по новой выполняем шаг алгоритма уже для нового остовного дерева.

Иначе, на цикле должна быть хотя бы одна вершина степени  $k - 1$  (Иначе, эти компоненты связаны - то есть это одна компонента, противоречие). Пометим все вершины цикла  $C$  как хорошие и объединим все вершины цикла и все компоненты, которые навешаны на вершины цикла в одну новую компоненту. Далее, снова ищем ребро между разными компонентами... (рекурсивно)

**Лемма 2.** Любая вершина  $u$ , помеченная хорошей - может быть сделана не блокирующей (т. е.  $\deg(u) \leq k - 2$ ) через цепочку улучшений в подграфе компоненты связности вершины  $u$ .

*Док-во.* Доказательство по индукции по числу объединений в алгоритме выше.

База: 0 объединений. Хорошие - только те вершины, у которых степень не больше  $k - 2$ . Подходит.

Переход: Вершина  $u$  стала хорошей на каком-то шаге. Значит, она лежала на цикле  $C$ . Рассмотрим ребро, которое инициировало этот цикл. Сначала сделаем его концы неблокирующими (по индукции), а потом добавим инициирующее ребро и удалим какое-то ребро инцидентное  $u$ , лежащее на цикле  $C$ ). Теперь  $u$  - не блокирующая

□

**Итого,** наш шаг алгоритма либо найдёт последовательность улучшений, которая уменьшит  $|S_k|$  на 1 (Следует из леммы 2).

Либо найдёт блокирующее множество  $B$  - все вершины степени  $k - 1$ , которые остались "плохими". И следовательно по Теореме 1 верно, что наше остовное дерево подходит под критерий поиска ( $k \geq \Delta + 1$ ).

### Итоговая асимптотика всего алгоритма

В алгоритме будет не более  $O(n \log n)$  шагов. Каждый шаг может быть выполнен за полиномиальное время. Также можно выполнить каждый шаг за почти линейное время  $O(m\alpha(n))$ , используя структуру данных "Система непересекающихся множеств", чтобы объединять компоненты связности. ( $\alpha(n)$  - обратная функция Акермана). Итоговая асимптотика  $O(mn \log(n)\alpha(n))$

## 3 Выводы

Алгоритм реализован и был протестирован на разных тесткейсах.

- 1  $G(n=1000, p=0.75)$  - (95% -  $k=2$ ) (5% -  $k=3$ ) (Выборка 100 тестов)  
 $G(n=1000, p=0.5)$  - (36% -  $k=2$ ) (64% -  $k=3$ ) (Выборка 50 тестов)  
 $G(n=1000, p=0.1)$  - (100% -  $k=3$ ) (Выборка 50 тестов)  
 $G(n=1000, p=0.01)$  - (100% -  $k=3$ ) (Выборка 50 тестов)  
 $G(n=1000, p=0.008)$  - (77% -  $k=3$ ) (33% - Не связный граф) (Выборка 50 тестов)

Случайные графы с очень маленьком количеством рёбер зачастую не связаны, поэтому алгоритм для них неприменим. Для  $n = 1000$  и  $p \geq 1\%$ , прослеживается зависимость, что чем больше рёбер в графе, тем меньше искомая оптимальная степень, причём она почти всегда 2 или 3.